# Hattrick: Solving Multiclass TE using Neural Models

Abd AlRhman AlQiam
Purdue University

Zhuocong Li
Purdue University

Satyajeet Singh Ahuja
Meta

Zhaodong Wang
Meta

Ying Zhang
Meta

Sanjay G. Rao
Purdue University

Bruno Ribeiro
Purdue University

Mohit Tawarmalani
Purdue University

## Abstract

While recent work shows ML-based approaches are a promising alternative to conventional optimization methods for Traffic Engineering (TE), existing research is limited to a single traffic class. In this paper, we present Hattrick, the first ML-based approach for handling multiple traffic classes, a key requirement of cloud and ISP WANs. As part of Hattrick we have developed (i) a novel neural architecture aligned with the sequence of optimization problems in multiclass TE; and (ii) a variant of classical multitask learning methods to deal with the unique challenge of optimizing multiple metrics that have a precedence relationship. Evaluations on a large private WAN and other public datasets show Hattrick outperforms state-of-the-art optimization-based multiclass TE methods by better coping with prediction error – e.g., for GEANT, Hattrick outperforms SWAN by 5.48% to 19.3% across classes when considering the traffic that can be supported 99% of the time.

## CCS Concepts

• **Networks → Traffic engineering algorithms**; • **Computing methodologies → Machine learning**.

## Keywords

Traffic Engineering, Wide-Area Networks, Network Optimization, Machine Learning

**ACM Reference Format:**
Abd AlRhman AlQiam, Zhuocong Li, Satyajeet Singh Ahuja, Zhaodong Wang, Ying Zhang, Sanjay G. Rao, Bruno Ribeiro, and Mohit Tawarmalani. 2025. Hattrick: Solving Multiclass TE using Neural Models. In *ACM SIGCOMM 2025 Conference (SIGCOMM '25), September 8–11, 2025, Coimbra, Portugal.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3718958.3750470

## 1 Introduction

The efficient management of cloud and Internet Service Provider (ISP) Wide Area Networks (WANs) depends on optimizing traffic routing to meet diverse service level objectives. While centralized controllers have emerged as an effective solution for network-wide optimization, recent research has demonstrated the potential of Machine Learning (ML) for Traffic Engineering (TE), particularly in leveraging predicted traffic matrices and scaling to large networks.

While ML-based TE holds promise, existing research has focused primarily on optimizing a single class of traffic for a single metric. In reality, cloud and ISP WANs support multiple classes of traffic with diverse requirements, including latency-sensitive applications with critical timeliness constraints (e.g., search) and elastic applications that prioritize minimizing transfer times [15, 16, 18, 23].

A naive extension of existing ML-based TE to multiple classes might involve using a simple Multi-Layer Perceptron (MLP) with an additive loss function that prioritizes higher-priority traffic. However, we demonstrate that this approach falls short due to the need for a neural architecture that is better aligned with the task of multiclass TE and the limitations of weighted sum loss functions in capturing the nuances of TE.

This disparity between existing ML-based TE approaches and the complexity of real-world networks motivates our research. Our work investigates two key questions: (1) What principled design approaches enable effective ML-based TE systems for multiclass scenarios? (2) How does a ML-based TE scheme architected on these principles perform compared to traditional "predict, then optimize" approaches when handling multiple traffic classes?

**Contributions.** In this paper, we present **Hattrick**[1], a new neural architecture and optimization methodology designed to dynamically allocate resources across traffic classes while accounting for future demand uncertainty. As part of Hattrick, we make several contributions:

• **First**, we have designed a multi-stage neural architecture that features a series of recurrent units which sequentially optimizes traffic classes in order of priority similar to many optimization-based multiclass TE approaches [11, 16, 19]. Notably, this novel architecture enables later stages to refine routing decisions made for higher-priority traffic in earlier stages, allowing for more flexible accommodation of lower-priority traffic without compromising the performance of higher-priority traffic. This design is consistent with a recently proposed optimal multiclass TE strategy [19], which we demonstrate offers significant advantages over more conventional multiclass TE approaches [11, 16] that lack such fine-tuning capabilities.

• **Second**, Hattrick also distinguishes itself by prioritizing performance under prediction error, a critical aspect that has received limited attention in previous multiclass Traffic Engineering (TE) schemes [11, 15, 16, 19]. To address this, Hattrick is deliberately designed to optimize key performance metrics that capture the

[1]Code available at https://github.com/Purdue-ISL/Hattrick/

This work is licensed under a Creative Commons Attribution 4.0 International License.
*ACM SIGCOMM '25, Coimbra, Portugal*
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1524-2/25/09
https://doi.org/10.1145/3718958.3750470

potential consequences of prediction error, including link under-utilization and capacity violations. By seamlessly integrating a differentiable, class-aware flow-based simulator into its training process, Hattrick more effectively models these detrimental effects. This new approach leverages the unique opportunity afforded by ML-based schemes to incorporate such simulators, thereby enhancing the robustness and reliability of TE in the face of prediction uncertainty.

• **Third**, Hattrick addresses the challenge of optimizing multiple, potentially conflicting objectives across traffic classes. A natural approach is to employ multiple loss functions, one per class, reflecting their respective objectives. However, a challenge arises from potential conflicts between these different optimization objectives. Existing multi-task learning approaches for conflicting objectives [29, 34] assume equal task importance, an assumption that breaks down in TE where strict priority ordering exists between traffic classes.

To overcome this limitation, Hattrick introduces a gradient projection technique based on [42] that ensures updates for lower-priority objectives cannot degrade the performance of higher-priority traffic classes. Our approach demonstrates improved performance compared to conventional multi-task learning approaches, providing a more effective and efficient solution for ML-based TE in multiclass scenarios.

**Results.** We evaluate Hattrick using a multi-week dataset of a private WAN which supports multiple traffic classes, and using other publically available topologies and traffic data. Our results show that Hattrick outperforms classical optimization-based multiclass TE methods by better coping with prediction error for multiple predictors. For instance, for GEANT, the fraction of traffic that Hattrick can support 99% of the time is higher by 5.48% to 19.3% relative to SWAN depending on the priority class. The corresponding improvements over BEST_MC are as high as 11.6% for some classes. An ablation study confirms the importance of the key elements of Hattrick's design including its neural architecture, and its precedence-aware multitask learning approach. Hattrick achieves significant computation benefits for larger topologies – e.g., for a topology of 754 nodes, it reduces computation time by a 280X and 36X relative to BEST_MC and SWAN respectively.

## 2 Background and Motivation

### 2.1 Background

**Multiclass TE.** Network operators [11, 16, 18] classify their traffic into multiple categories depending on the timeliness requirements. For example, search traffic has high priority since it has stringent timeliness requirements, and video traffic may be given high priority since loss impacts user experience. In contrast, background traffic (e.g., replicating data across data centers) may be assigned a lower priority. Each traffic class is associated with a Service Level Objective (SLO) which indicates a minimum percentage of time that both network connectivity and promised bandwidth must be available between a pair of sites. The SLOs are stringent – production networks have SLO requirements ranging from 99.99% (4 9's) for the highest priority class, to 99% (2 9s) for lower priority classes [17].

While production networks may maintain any number of traffic classes, throughout this paper we will assume three classes by default for concreteness (following [16]). The TE pipeline takes as inputs a network topology, a predicted traffic matrix per class (i.e. one for high, one for medium, and one for low), a set of link capacities, and a set of precomputed tunnels (paths). Traffic matrices are predicted from past measurements of traffic matrices (typically collected in the granularity of minutes). The TE algorithm then determines how each class's traffic should be routed on the tunnels while optimizing network-wide metrics related to the traffic of each class.

**ML-Driven TE.** Tradtionally, TE algorithms are based on optimization methods, whose effectiveness is sensitive to the accuracy of traffic matrix predictions. Robust optimization techniques (e.g., [5, 37, 43]) consider a set of potential traffic matrices, and optimize the worst-case performance across all the matrices. However, explicitly specifying all matrices is non-trivial, and the performance can be overly conservative for any given matrix [7, 24] (we discuss further in §6).

Recent studies [3, 26, 32, 40] have investigated the application of machine learning (ML) to TE, where ML models directly determine traffic routing decisions along tunnels. These schemes are based on neural models which decide how traffic must be routed in the future while taking as input past matrices [32] or predictions based on past matrices [3]. Neural models for WAN TE are promising given they can learn to account for potential prediction inaccuracies [3, 32] unlike conventional optimization methods. Additionally, ML-based TE has been shown to offer reduced computation times for larger-scale topologies [40]. While ML-driven TE holds promise, existing schemes are limited to a single class of traffic. In contrast, real-world TE scenarios typically involve multiple traffic classes, which is the primary focus of this paper.

### 2.2 Challenges for ML-based multiclass TE

In designing ML architectures for multiclass TE, two key challenges must be addressed which do not arise with single-class TE:

**Model a sequence of optimization problems.** Multiclass TE entails solving a series of interconnected optimization problems, rather than a single problem. This is exemplified in [16], where the total flow for high-priority traffic is first optimized, followed by the optimization of the total flow for the next traffic class in the residual network, and so on. More formally, a multiclass TE problem can be characterized as a sequence of $K$ optimization problems, where $K$ denotes the number of distinct traffic classes, and each subsequent problem is solved over the residual network obtained from the previous optimization step.

**Capture multiple objectives with precedence relationships.** In contrast to single-class TE, which often focuses on a single objective, multiclass TE involves optimizing multiple, hierarchically-related metrics. In many settings [16], the primary objective is to maximize the flow for the highest-priority traffic class, while secondary and tertiary objectives aim to maximize the flow for subsequently lower-priority classes, subject to the precedence relationships between them. This hierarchical structure reflects the inherent priority ordering among traffic classes, where the optimization of higher-priority classes takes precedence over that of lower-priority ones.

**Why simple extensions are insufficient?** To illustrate the challenges in adapting single-class ML-based TE schemes to multiclass TE, we consider DOTE [32], a prominent and influential ML-based TE approach. DOTE takes as input the last $H$ traffic matrices and predicts the optimal routing allocations for each tunnel in the next time step. Built on a fully connected neural network (DNN), also known as a multilayer perceptron (MLP), DOTE supports various common single-class TE optimization objectives, including maximizing the flow carried, and minimizing maximum link utilization (MLU).

We extended DOTE to multiclass TE by modifying its input to include $3H$ traffic matrices, representing the past $H$ time steps for each of the three traffic classes. The same MLP architecture was retained, but the output was adapted to generate allocations for each flow and traffic class on each tunnel. Furthermore, we modified DOTE's loss function to maximize a weighted total flow metric. The metric computes the weighted sum of traffic carried across all classes, with different weights assigned to each of the three classes to reflect their relative priorities.

However, this simple extension falls short and fails to adequately capture the intricacies of multiclass TE. To see this, consider Figure 1 which presents an evaluation of the modified DOTE on the GEANT topology. The figure presents a series of subplots, each corresponding to a distinct weight configuration in the additive loss function. Each figure plots the FulfillRatio for each of the three traffic classes (the fraction of traffic carried successfully for each class, and we elaborate further on the metric in §3.)

To provide a meaningful baseline, the results are normalized against the optimal flow, computed using an oracle with perfect knowledge of ground truth future traffic matrices (see §4 for a detailed description of the methodology). Note that normalized values may exceed 1 for lower classes, indicating a priority inversion in which higher class traffic receives a suboptimal allocation, which leaves more bandwidth for lower classes to utilize. As an extreme example, a sub-optimal scheme that drops all high class traffic, and uses the entire capacity for lower priority traffic will achieve zero NormFulFill for higher priority, and a NormFulFill that may exceed 1 for lower priority. We further expand with an example in Appendix A.

A close examination of Figure 1 shows the inadequacies of the augmented DOTE in reconciling the competing demands of heterogeneous traffic classes. The introduction of weighted losses, notwithstanding its intuitive appeal, fails to strike a balance between the diverse objectives of multiple traffic classes, regardless of the chosen weight configuration.

The root of this issue lies in the inherent tension between high-priority and low-priority traffic. When higher weights are assigned to high-priority traffic (as evident in the top graphs), its performance improves, but at the cost of degraded performance for lower-priority classes. This is not merely a matter of trade-off, but rather a more fundamental limitation: our analysis reveals that gradient dominance, induced by the heavily-weighted high-priority class, stifles parameter updates that could potentially benefit lower-priority traffic without hurting higher-priority ones. In contrast, adopting equal weighting (bottom right) appears to improve the performance of lower-priority classes, but this comes at the expense of compromised performance for high-priority traffic. This phenomenon underscores the complexity of balancing competing objectives in multiclass TE and highlights the need for a more sophisticated approach that can navigate these challenges.

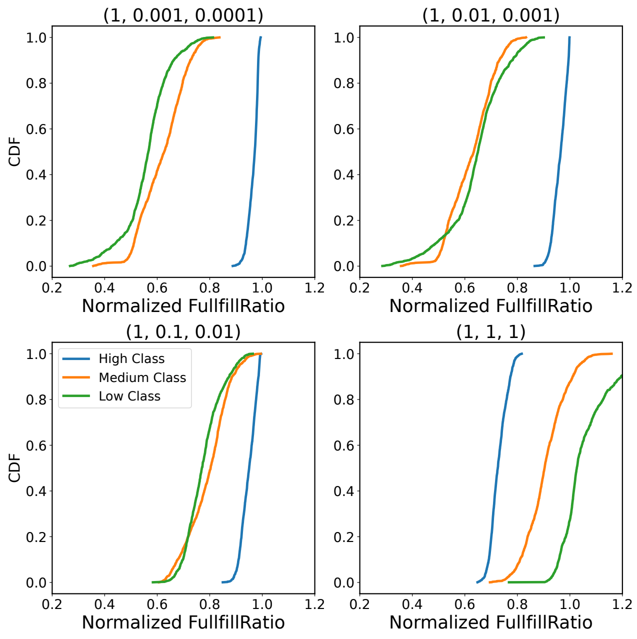


**Figure 1: Performance of DOTE augmented to support multiple classes with additive loss on the GEANT topology. Each subplot shows the NormFulFill achieved for the three traffic classes under a different additive weight configuration, with the tuple below indicating the relative weights assigned to high, medium, and low priority traffic classes, respectively. The results highlight the challenge of using simple extensions of single class ML-based TE methods, and using additive loss. Normalized values exceed 1 in the bottom right figure because suboptimal performance for high priority traffic, leaves more capacity that may be utilized by lower priority traffic.**

**Summary.** Our analysis suggests that the extended DOTE approach falls short due to two fundamental limitations: (i) the weighted loss function proves ineffective in reconciling the competing demands of diverse traffic classes, leading to suboptimal performance; and (ii) the straightforward DNN architecture lacks the necessary expressiveness to co-optimize different traffic classes. The remainder of this paper presents our approach, Hattrick, which explicitly addresses these two issues, and demonstrates the importance of tackling each of them in §4.

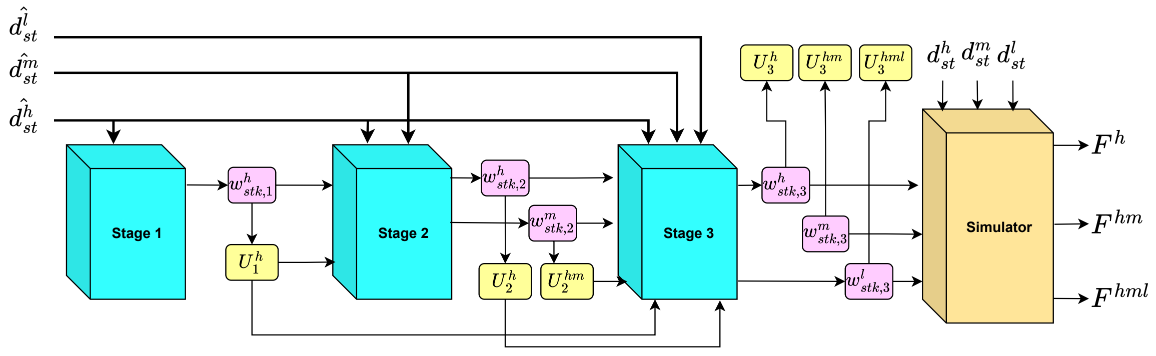


**Figure 2: Overview of Hattrick's multistage architecture depicting three priority classes ($h, m, l$).**

## 3 Hattrick Design

We introduce Hattrick, a novel neural network architecture designed for the complexities of multiclass TE optimization. Hattrick is grounded in two key principles:

**1. Multi-stage sequential modeling of multiclass TE optimization problems.** Hattrick's neural architecture is explicitly designed to capture the inherent sequentiality of multiclass TE optimization problems, mirroring the relationships between optimization variables and constraints. This multi-stage architecture is described in detail in §3.1.

**2. Precedence-aware multitask learning.** Building on recent advances in multitask learning, we adapt these techniques to accommodate the strict precedence relationships that should govern the optimization of metrics across different traffic classes. Unlike conventional multitask learning approaches, which assume a flat importance structure among tasks (e.g., [29, 34]), Hattrick incorporates a gradient projection method that prioritizes updates based on class precedence. This ensures that the optimization process respects the hierarchical relationships between traffic classes, preventing lower-priority classes from compromising the performance of higher-priority ones. We elaborate on this approach in §3.3.

### 3.1 Hattrick's multi-stage architecture

**Modeling optimal multiclass TE routing.** A promising approach to tackling multiclass TE could emulate the sequential optimization process employed by traditional methods, such as SWAN [16] and EBB [11]. This process typically unfolds in three stages: (i) optimizing for higher-priority traffic; (ii) creating a residual network with updated link capacities that reflect the remaining capacity after allocating higher-priority traffic; and (iii) routing lower-priority traffic in the residual network.

However, this approach is not without its limitations. Since multiple routing configurations for high-priority traffic can yield the same flow allocation, prematurely pinning the routing in the first stage unnecessarily constrains the flexibility for lower-priority traffic classes, ultimately impacting their allocations. Instead, the optimal strategy—an approach that has received little attention in the literature, with only tangential mention in a recent work [19])—only decides the allocation to higher priority traffic in earlier stages, but defers the actual decision on how this traffic must be routed to later stages, as long as the earlier stage allocations are honored. We describe this scheme further in §4.

The Hattrick architecture is inspired by this optimal strategy, which enables Hattrick to explore a wider range of possible solutions and improve overall performance. The architecture of Hattrick comprises multiple stages that sequentially consider different priority classes, as shown in Figure 2. The first stage focuses on optimizing decisions for high-priority traffic. While each subsequent stage focuses on its corresponding class, adaptive adjustments may be made to earlier stage decisions. These adjustments are designed to improve the performance of lower priority without hurting higher priority traffic, as we elaborate later.

**Learning to cope with prediction error.** Prior literature on multiclass TE [11, 16, 19] does not extensively discuss the impact of prediction errors, a key focus of Hattrick. These schemes typically solve MaxFlow problems. While MaxFlow is an ideal metric when predictions are perfect, it falls short when predictions are imprecise. For example, if flow allocations are limited based on predicted traffic, the allocation may be unduly conservative when actual traffic is higher. Further, strategies that are feasible for the predicted matrix (e.g., if the fraction of traffic of each flow is limited) are not guaranteed to be feasible upon traffic realization (say, when the actual traffic is higher). Real-world WANs accommodate such violations through WAN router queuing policies that prioritize higher class traffic, so losses typically only correspond to lower class traffic [16, 18].

Instead, Hattrick seeks to optimize **FulfillRatio**, i.e., the fraction of traffic in each class that is delivered successfully *after* accounting for the impact of link capacity violations caused by prediction errors. To capture these effects, we employ a differentiable, class-aware flow-based simulator, whose details are provided in §4. During training, Hattrick leverages this simulator to compute the FulfillRatio metrics, which are then incorporated into its loss function. Learning in such a fashion is a unique opportunity available only to ML-based schemes, as modeling such artifacts is difficult in conventional optimization methods.

**Aiding learning with auxiliary metrics.** Production networks are typically sufficiently provisioned to handle traffic from the most critical classes with ease, resulting in a large solution space that can

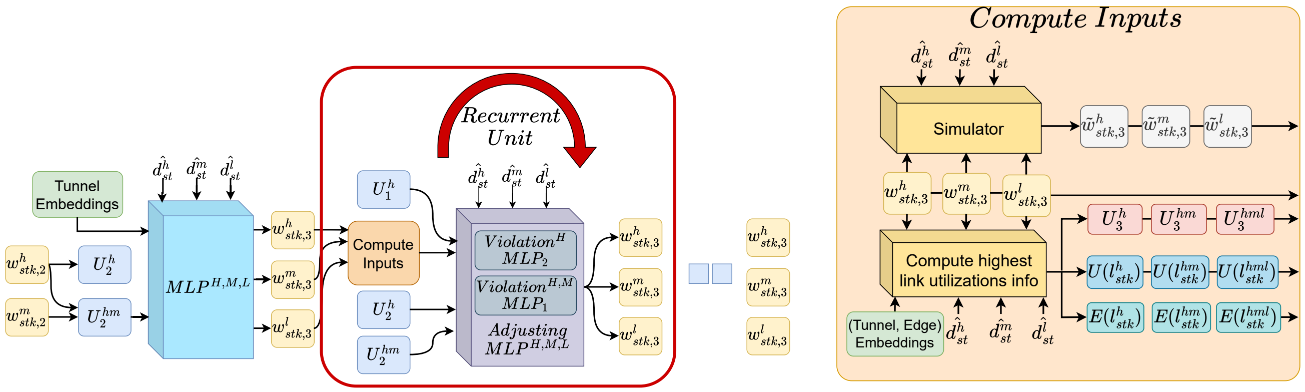


**Figure 3: Details of Hattrick's architecture expanding the third stage**

| | |
|---|---|
| $\hat{d}^c_{st}$ | Predicted traffic of class $c\{h, m, l\}$ from $s$ to $t$ |
| $d^c_{st}$ | Ground truth traffic of class $c$ from $s$ to $t$ |
| $w^c_{stk,i}$ | Split ratios, i.e., frac of traffic of class $c$ from $s$ to $t$ routed on tunnel $k$ after stage $i$ $\forall c, s, t, i \sum_k w^c_{stk,i} = 1$ |
| $U^h_i$, $U^{hm}_i$ $U^{hml}_i$ | MLU (Util of most congested link) after stage $i$ only considering the top class, top two classes or all classes |
| $F^h$, $F^{hm}$ $F^{hml}$ | FulfillRatio for traffic of the top class, top two classes, and all traffic |

**Table 1: Notation table**

achieve high FulfillRatio for these classes. However, this leads to a challenging optimization problem for ML-based schemes, as loss functions based solely on these metrics yield limited learning over these top classes due to vanishing gradients. Specifically, solutions may saturate links with higher priority traffic, which is a challenge when this traffic is more than predicted. Instead, Hattrick considers multiple link utilization metrics (summarized in Table 1) in its loss functions in addition to FulfillRatio. This steers learning towards preferring lower utilization solutions when multiple candidates with high FulfillRatio exist.

## 3.2 Architecture details

Formally, we adopt the notation presented in Table 1, where $\hat{d}$ variables represent predicted traffic, and $w$ variables denote split ratios (fractions of traffic sent on different tunnels) determined by Hattrick at each stage. As illustrated in Figure 2, the first stage computes an initial set of split ratios for high-class traffic ($w^h_{stk,1}$) that minimize $U^h_1$, the utilization of the most congested link (or Maximum Link Utilization, MLU) when only high-class traffic is considered.

The second stage then determines split ratios for medium-class traffic ($w^m_{stk,2}$) with the objective of minimizing $U^{hm}_2$, the MLU when traffic from the top two classes is considered. This stage is also allowed to adjust the split ratios of high-class traffic (with $w^h_{stk,2}$ denoting the updated values). The final stage aims to minimize $U^{hml}_3$, the MLU when all traffic is considered, while ensuring that the $U^h$ and $U^{hm}$ metrics are not compromised. This is accomplished by solving a multi-objective optimization problem that balances the competing goals of minimizing $U^{hml}_3$ and maintaining the performance of the higher-priority classes.

Figure 3 presents a more detailed view of the third stage alone (the other stages are similar and omitted). Consider that the previous stages have generated initial candidate split ratios for the high and medium classes ($w^h_{stk,2}$ and $w^m_{stk,2}$) and computed the resulting MLU variables ($U^h_1$, $U^h_2$ and $U^{hm}_2$). The third stage comprises (i) a multilayer perceptron (MLP) that fine-tunes variables $w^h$ and $w^m$ and proposes an initial split ratio for the lower priority class ($w^l$); and (ii) a recurrent unit, which further fine-tunes $w^h$ and $w^m$ and improves $w^l$.

The recurrent unit is executed per tunnel and includes a computation block (depicted in Figure 3 (right)) which we describe next. First, we observe that the $w$ variables assume that all traffic of every flow of each class is routed. To capture capacity violations, we use the same class-aware flow-based simulator described earlier to compute corresponding $\tilde{w}$ variables which denotes the effective traffic of each flow successfully delivered after making adjustments for capacity violations. The recurrent unit takes as input both $w$ and $\tilde{w}$ variables. In addition, the recurrent unit takes as input a set of variables computed for three cases: (a) only considering traffic of the top class; (b) considering traffic of the top two classes; and (c) considering all traffic. Let $l^h_{stk}$ denote the bottleneck (most congested) link of the $k^{th}$ tunnel from $s$ to $t$ when only high priority traffic is carried. The variables corresponding to the case (a) include (i) a suitable embedding based of this bottleneck link E($l^c_{stk}$) which we elaborate further later; (ii) the utilization of the bottleneck link $U(l^c_{stk})$; and (iii) the network-wide MLU when only high priority traffic ($U^h_3$).

A primary goal of the recurrent unit is to fine-tune the $w^h$ and $w^m$ variables to accommodate lower priority traffic without compromising the $U^h$ and $U^{hm}$ metrics computed in the previous stage. To achieve this, we employ an MLP (Adjusting MLP$^{H,M,L}$) whose actions are supervised by another neural network, Violation-MLP, through a custom loss function (detailed in §3.3) that strikes a balance between optimizing $U_3^{hml}$ and preserving the $U^h$ and $U^{hm}$ metrics. In cases where Violation-MLP deems the adjustments unsatisfactory, it intervenes by modifying the behavior of the Adjusting MLP, thereby striving that the higher priority $U^h$ and $U^{hm}$ metrics are preserved.

Finally, as illustrated in Figure 3, Hattrick leverages tunnel and edge embeddings as inputs, building upon the methodology outlined in previous work [3]. Specifically, this process involves two key components: (i) a graph neural network that ingests the network topology along with capacity information, ultimately generating a rich embedding of edges; and (ii) a set transformer that uses these edge embeddings to produce embeddings of tunnels. Furthermore, the set transformer also yields tunnel-edge embeddings, which represent the embeddings of each edge conditioned on the specific tunnel it belongs to, thereby capturing the relationships between tunnels and their constituent edges. We implemented Hattrick using PyTorch [31] and used Adam [21] optimizer to optimize its weights.

## 3.3 Loss function and learning algorithm

During the training phase, Hattrick generates the $w^c_{stk,3}$ variables that determine the traffic routing over the tunnels. Although these variables may result in link capacity violations, they are used as inputs to the class-aware flow simulator, alongside the ground truth demand matrices ($d^c_{st}$). This setup enables the computation of key metrics, including $F^h$, $F^{hm}$, and $F^{hml}$, which are essential for the optimization process.

We note that during inference, Hattrick produces the $w^c_{stk,3}$ variables without relying on ground truth demand matrices. Instead, it operates solely based on the learned patterns and relationships, routing traffic according to these generated variables.

For its training, Hattrick has six distinct objectives with six distinct loss functions, organized in priority order: (-$F^h$, $\max_i U_i^h$, -$F^{hm}$, $\max_i U_i^{hm}$, -$F^{hml}$, $\max_i U_i^{hml}$). These loss functions are structured to reflect the system's multiple objectives, with the first two focusing on the highest priority class, the next two on the aggregate traffic of the top two classes, and the final two on all traffic. Each pair of loss functions within these categories includes both the FulfillRatio metric (e.g., $F^h$, $F^{hm}$, $F^{hml}$) and the utilization metric (e.g., $U_i^h$, $U_i^{hm}$, $U_i^{hml}$). The utilization metrics are formulated to minimize the maximum value across stages, denoted by $\max_i U_i^h$, ensuring that Hattrick keeps it as small as possible while adjusting variables like $w_h$. Similarly, $\max_i U_i^{hm}$ indicates the maximum of the $U^{hm}$ metric across stages, which Hattrick again tries to keep as small as possible, and so on.

To effectively manage the precedence relationships between objectives, we introduce a gradient projection approach that tackles two key challenges: (i) reconciling conflicting gradients arising from distinct objectives and (ii) maintaining task precedence during the optimization process. Our method operates as follows: after completing the forward pass, computing losses, and obtaining gradients for a given mini-batch, we update the model parameters by projecting the gradients of lower-priority tasks onto the orthogonal

**Function 1** Parameter Update with Gradient Projection

```
    Input: $\theta,\ \eta,\ g_1,\ g_2,\ g_3$
1:  $\mathbf{g}_2' = \mathbf{g}_2 - \frac{\mathbf{g}_2 \cdot \mathbf{g}_1}{\mathbf{g}_1 \cdot \mathbf{g}_1}\mathbf{g}_1$
2:  $\mathbf{G} = [\mathbf{g}_1 \quad \mathbf{g}_2']$,
    $\mathbf{g}_3' = \mathbf{g}_3 - \mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{g}_3$
3:  $\mathbf{g} = \mathbf{g}_1 + \mathbf{g}_2' + \mathbf{g}_3'$
4:  $\theta \leftarrow \theta - \eta * \mathbf{g}$
```

subspace of higher-priority task gradients. This deliberate projection eliminates the components of lower-priority gradients that could potentially interfere with the optimization of higher-priority tasks, thereby ensuring that the model prioritizes the optimization of high-priority flows without compromising their performance. By doing so, the model can focus on optimizing lower-priority flows in a manner that respects and preserves the optimization objectives of higher-priority flows.

Specifically, let $g_1, \dots, g_n$ denote the gradients of loss functions $\text{loss}_1, \text{loss}_2, \dots, \text{loss}_n$, respectively. Given these gradients, model parameters $\theta$, and learning rate $\eta$, our approach (detailed in Function 1 for the case of $n = 3$ metrics) uses a multi-step projection process to enforce task precedence. For each $i = 2, \dots, n$, we project the gradient $g_i$ onto the null space of $\text{span}(g_1, \dots, g_{i-1})$, yielding $g_i'$. This projection ensures that optimizing $\text{loss}_i$ does not compromise the optimization of higher-priority losses $\text{loss}_1, \dots, \text{loss}_{i-1}$. By applying this projection in every mini-batch, Hattrick effectively prioritizes the optimization of higher-importance metrics, shielding them from potential interference by lower-priority metric gradients. As shown in §5.2, this precedence-aware multi-task learning approach outperforms optimizing a traditional weighted sum of all objectives.

The above design can be simplified in practice. In many real-world scenarios (including our experimental settings), networks are sufficiently provisioned to handle high and medium traffic demands. As a result, the gradients of the $F^h$ and $F^{hm}$ metrics become negligible, having an insignificant impact on the training of Hattrick parameters. Consequently, these metrics can be safely omitted from the design without compromising its effectiveness, thereby streamlining the optimization process.

While we do not explore, it is feasible to extend Hattrick to consider latency requirements. For instance, the Hattrick framework may be used with latency as a secondary metric, and FulfillRatio as the primary metric for each class, or using a weighted metric which maximizes throughput while penalizing latency following [16]. It is also possible to extend Hattrick's implementation to restrict more latency-sensitive higher priority traffic to a smaller subset of lower latency tunnels while allowing all tunnels for other classes. We defer a detailed investigation of these approaches to future work.

## 4 Evaluation Methodology

Our primary goal is to evaluate the effectiveness of Hattrick for multiclass TE under prediction error by comparing it with state-of-the-art multiclass TE schemes. We also conduct an ablation study to investigate the impact of individual design components on Hattrick's performance, allowing us to quantify the importance of each mechanism.

**Datasets.** We conducted experiments with PrivateWAN, a real-world private WAN with several tens of nodes and several hundreds of edges. The dataset spans multiple weeks, and records at a per second granularity (i) the topology (nodes, edges, and link capacities); and (ii) three traffic matrices corresponding to three classes (high, medium, and low) supported by the network. Across snapshots, high priority traffic is 23%-42% of the total traffic, while medium priority traffic is 21%-38%, and low priority traffic is 25%-46%. The PrivateWAN topology is dynamic, exhibiting temporal variations characterized by fluctuations in the set of nodes and links (owing to maintenance, failures, and organic growth), and significant changes in link capacities (owing to partial and full link failures), as well as periodic tunnel recomputations on significant changes.

We also conduct experiments with GEANT [1, 35]. While publicly available traffic matrices exist for GEANT, splits by traffic class are not (in fact, we are unaware of any publicly available dataset which provides matrices corresponding to different classes). Hence, we split the traffic of GEANT into three classes using traffic ratios derived from the PrivateWAN data. We complement the above datasets with larger topologies from the Internet Topology Zoo [22] where we use synthetic traffic matrices.

**Schemes compared.** We compare Hattrick, an ML-based TE scheme with two state-of-the-art multiclass TE schemes: SWAN[16] and BEST_MC[19]. Further, while existing ML-based TE schemes do not consider multiple classes, we have conducted experiments comparing Hattrick with DOTE and HARP (two recent and representative single class ML-based TE schemes), augmenting them in natural ways to support multiple classes. We have also compared Hattrick to several of its variants to explore the importance of its individual design components. We discuss these extensions to DOTE and HARP, and Hattrick variants in §5.2, and elaborate on SWAN and BEST_MC below.

**SWAN.** Given the three traffic matrices, SWAN [16] first i) solves a MaxFlow optimization problem considering high priority traffic alone on the original network to determine the allocation and routing for high priority traffic; (ii) creates a residual network reflecting the remaining capacity after allocating and routing high priority traffic; and (iii) solves a MaxFlow optimization problem in the residual network and considering the traffic of the next class alone to decide the allocation and routing for that class. Steps (ii) and (iii) are performed iteratively until all classes are handled.

**BEST_MC.** BEST_MC [19] shares the same Step (i) as SWAN to maximize the total flow of high priority traffic. However, it only uses allocation decisions at this step, and does not freeze how higher priority traffic is routed. Instead, in the next step, it solves a MaxFlow problem for the intermediate traffic class while adding a constraint that the total flow of high priority traffic must match Step (i), yet allowing high and intermediate class traffic to be routed flexibly as long as capacity constraints are met. Finally, it optimizes the total flow for the lowest priority class while adding constraints such that the total flow of the high and intermediate traffic classes match the previous steps. Note that BEST_MC is optimal when given ground-truth matrices, but in our results, it is computed using predicted matrices. We present the full formulation of BEST_MC in the Appendix.

**Methodology and metrics.** All schemes that we compare take three predicted traffic matrices (corresponding to the three classes) as input and generate split ratios (the fraction of traffic to be sent on each tunnel for each pair and each class) for the future unknown traffic matrix. We evaluate two different prediction methods: (i) Linear Regression; and (ii) Exponential Smoothing. All our experiments use the K shortest paths as tunnels for all node pairs and all traffic classes. We use K=15 for PrivateWAN, K=8 for GEANT and UsCarrier [22], and K=3 for KDL [22]

Although the split ratios generated by a system maybe feasible for the predicted traffic matrix, link capacities may be violated on the ground truth matrix if actual traffic exceeds predicted. In practice, real-world WANs handle such scenarios through router queuing policies that prioritize traffic of higher classes [16]. We capture this with a **class-aware flow level simulator** that we describe next.

Given (i) split ratios generated on a predicted matrix; and (ii) a ground truth matrix, the simulator computes the traffic for each flow of each class which is successfully carried. The simulator does so iteratively, with the first iteration considering high priority traffic alone, and the $k^{th}$ iteration considering the $k^{th}$ traffic class. In the first iteration, the simulator starts by computing utilizations of each link considering ground truth traffic of the highest priority class using the split ratios for flows of that class suggested by a scheme. Next, for each tunnel, the simulator computes the utilization $u$ of the most congested link of that tunnel and reduces the allocation to the tunnel by a factor of $u$ if $u > 1$. At the conclusion of the first iteration, no link's capacity is violated. The resulting allocation to each flow is considered to be the amount of traffic successfully delivered for that flow. The $k^{\text{th}}$ iteration is similar except that (i) the simulator uses the residual topology with link capacities adjusted to account for the flow allocated to previous classes; and (ii) adjustments are made to the allocations of the $k^{\text{th}}$ traffic class to account for link capacity violations if any.

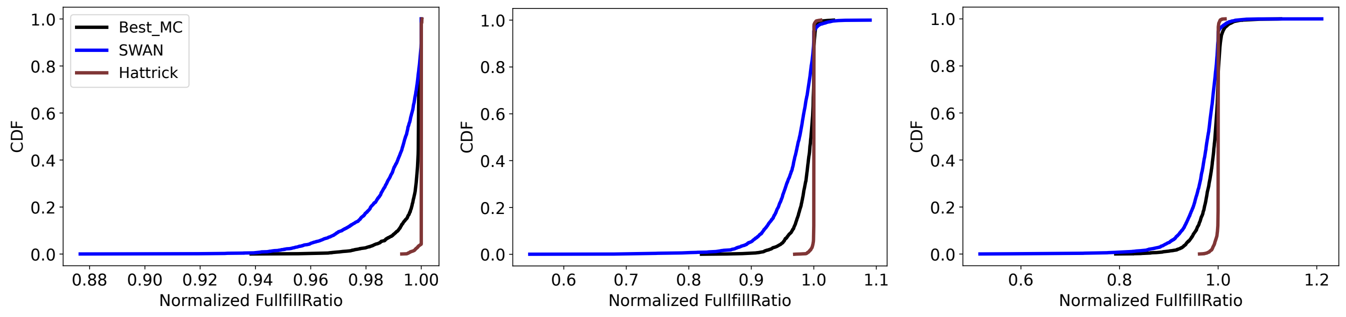
For each traffic class, the simulator reports the *Fulfillment Ratio* for that class which is the fraction of total traffic of that class which is successfully carried. We normalize the fulfillment ratios relative to an oracle which performs optimal multiclass TE with knowledge of ground truth. Thus, we report the *Normalized FulFillment Ratio* for each class (which we abbreviate as **NormFulFill**). For high priority traffic, NormFulFill is at most 1. For other traffic classes, a NormFulFill may exceed 1 with a scheme which indicates a priority inversion with that scheme – i.e., the scheme carried more traffic of the class than the optimal multiclass TE scheme at the expense of higher priority traffic class.

## 5 Results

We begin by comparing Hattrick to state-of-the-art multiclass TE schemes (SWAN and BEST_MC) in §5.1 using PrivateWAN and GEANT. Next, we show the benefits of the design elements of Hattrick in §5.2. We evaluate Hattrick's benefits with larger topologies in §5.3, and its performance on other predictors in §5.4.

### 5.1 Hattrick vs Current Multiclass TE

We compare Hattrick to the state-of-the-art multiclass TE schemes SWAN and BEST_MC for the PrivateWAN and GEANT networks.



**(a) High-class only** **(b) Med-class only** **(c) Low-class only**

**Figure 4: Comparing Hattrick to SWAN and BEST_MC using GEANT dataset considering exponential smoothing predictor. All schemes are normalized to an optimal oracle that has ground truth information. Normalized values for a scheme may exceed 1 for lower priority classes owing to a priority inversion (the scheme may under-perform for a higher priority class and use the capacity for a lower priority class).**

In both cases, we trained Hattrick using about 6000 samples, used around 1500 additional samples for validation, and the rest for testing (about 21000 samples for PrivateWAN, and around 2700 samples for GEANT).

**GEANT.** Figure 4 compares the performance of the three systems over the three traffic classes for the GEANT topology. The figure shows that Hattrick is close to optimal for all classes and clearly out-performs SWAN and BEST_MC. For instance, Hattrick achieves a 10%ile NormFulFill of 1.0, 1.0 and 0.9986 for the three classes. In contrast, the corresponding values for BEST_MC are only 0.9915, 0.9652 and 0.9552, and for SWAN are 0.9713, 0.9194 and 0.9264. The benefits are even stronger at the tail – for example Hattrick achieves a 1%ile NormFulFill (which corresponds to two 9's of performance) of 0.9963, 0.9921 and 0.9824 for the three classes while SWAN performs significantly worse achieving 0.9445, 0.832, and 0.8229, and BEST_MC achieves 0.971, 0.9186, and 0.8924 respectively for the three classes.

These results demonstrate Hattrick's ability to outperform methods based on optimization, given ML-based TE schemes can learn to compensate for prediction errors. Further, SWAN does significantly worse than BEST_MC because it makes decisions on how to route higher priority traffic without accommodating the needs of lower priority traffic. In contrast, Hattrick may adjust split ratios for higher priority traffic as long as its performance is unaffected to aid other classes.

**PrivateWAN.** Figure 5 compares the systems for PrivateWAN. First, we note that all schemes perform better in PrivateWAN relative to GEANT – this is because PrivateWAN is a better provisioned network with link utilizations lower overall. Second, Hattrick is close to optimal for all three classes, and continues to provide benefits over other schemes. For lowest priority traffic, Hattrick achieves a 10*%ile* NormFulFill of 0.996 while the corresponding values for SWAN and BEST_MC are 0.924 and 0.967 respectively. For medium priority traffic, SWAN performs significantly worse than Hattrick with a 10*%ile* NormFulFill of 0.940, while Hattrick achieves 0.996. Finally, for high priority traffic, Hattrick improves tail performance over both SWAN and BEST_MC. Overall, the results highlight Hattrick's ability to provide benefits over classical optimization schemes.

## 5.2 Benefits of Hattrick's design elements

We start by demonstrating the benefits of two key innovations with Hattrick: (i) a new neural architecture aligned with multiclass TE; and (ii) a new precedence-aware multitask learning algorithm. To this end, we consider a variant of Hattrick which we term Hattrick-AL. This scheme has the same architecture as Hattrick but uses a weighted additive loss function (with larger weights for higher priority classes) instead of the learning algorithm discussed in §3.3. We then demonstrate the benefits of Hattrick's alignment approach which allows tuning high priority traffic to accommodate the needs of lower priority traffic over an alternative that does not. We conclude with an ablation that evaluates the benefits of incorporating a simulator in the training of Hattrick.

**Benefits of architecture.** To evaluate the benefits of Hattrick's architecture, we compare Hattrick-AL with a variant of DOTE which we introduced in §2 on GEANT. Recall that this scheme (which we refer to as DOTE-MC) is our augmentation of DOTE [32] to support multiple classes by using a simple DNN with a weighted additive loss function. Figure 6 compares DOTE-MC and Hattrick-AL for a weight configuration where DOTE-MC performed relatively better for the GEANT topology. Each pair of boxplots corresponds to a traffic class, and show the distribution of NormFulFill for the two schemes. Hattrick-AL is clearly better for all classes, with particularly strong benefits for the medium and lower priority classes. These results show the benefits of using Hattrick's architecture over a simple MLP.

**Benefits of precedence-aware learning.** Next, we compare Hattrick and Hattrick-AL where the architectures are the same, but the schemes differ in that Hattrick uses a precedence-aware multitask learning algorithm, while Hattrick-AL uses an additive loss function. Figure 7 compares the schemes for the PrivateWAN topology. The figure shows that Hattrick performs much better on low priority traffic – the median and 10%ile NormFulFill of Hattrick-AL is 0.97 and 0.917, while the corresponding values are 1.00 and 0.996 for Hattrick. The systems perform comparably on high and medium priority traffic – we omit results for brevity. Overall, the results show that besides architecture, the learning algorithm used by Hattrick is important to its performance.

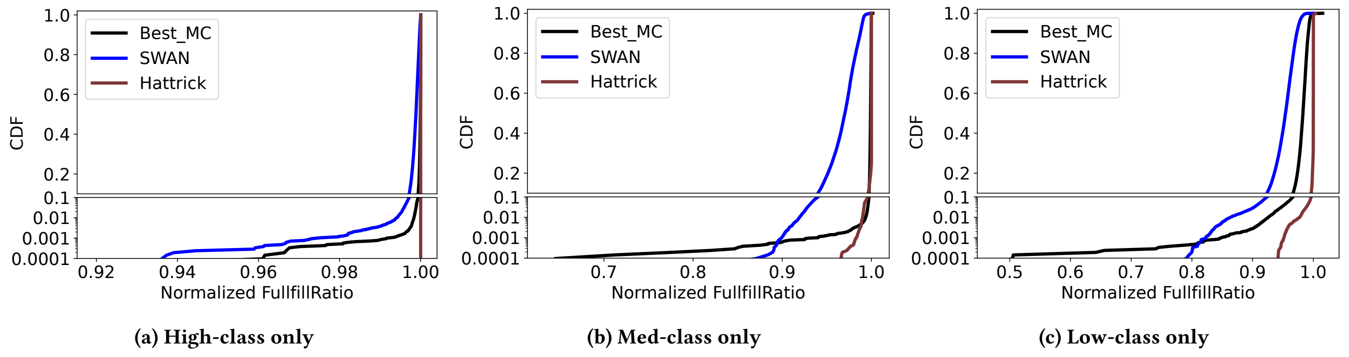


**Figure 5: Comparing Hattrick to SWAN and BEST_MC using PrivateWAN dataset. We show a log scale for the Y-Axis at lower percentiles to better highlight tail performance.**

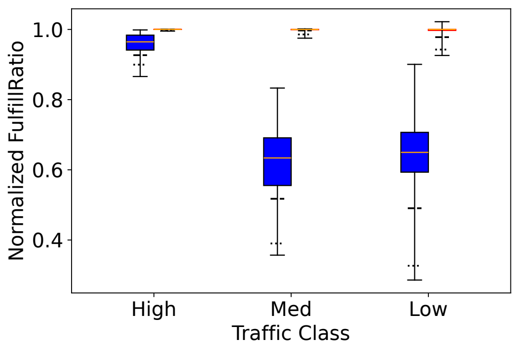


**Figure 6: Benefits of Hattrick's architecture by comparing DOTE-MC (left) and Hattrick-AL (right) on GEANT. Both schemes use an additive loss function with the set of weights of (1, 0.01, 0.001) assigned to high, medium, and low priority traffic classes, respectively. The dashed and dotted lines show 1%ile and 10%ile, respectively.**

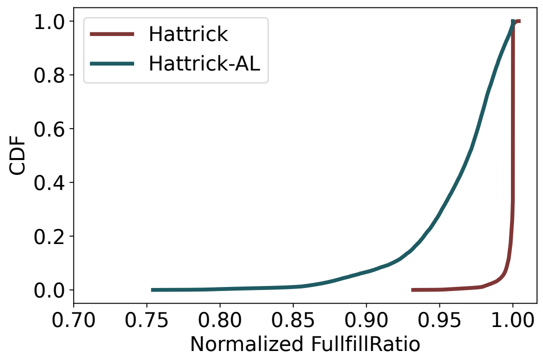


**Figure 7: Benefits of precedence learning. Comparing Hattrick (which uses precedence-aware multitask learning) and Hattrick-AL (with additive loss multitask learning) on PrivateWAN for low priority traffic.**

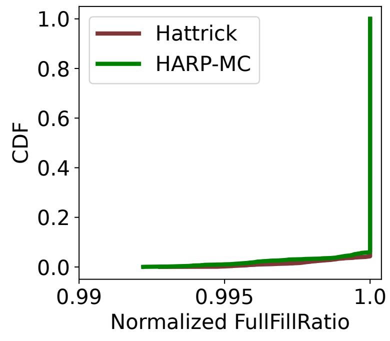
**Benefits of Hattrick's alignment approach** As discussed in §3, Hattrick's architecture allows fine-tuning of how higher priority traffic is routed in later stages while preserving performance for lower priority traffic. We will compare this design decision of Hattrick with an alternate ML-based approach that does not support such fine-tuning.

To this end, we extend HARP [3], a recent neural TE scheme designed for traffic of a single class, to support multiclass TE. We do so by training separate HARP models for each traffic class as follows: (i) a first network trained on the original topology and higher priority traffic; and (ii) a second network trained on the residual topology (with the remaining capacity) and medium priority traffic; and (iii) a third network trained on the residual topology and low priority traffic. Note that link capacities in the residual topology can exhibit substantial variability across examples depending on the demands of the higher-priority class in each instance. We focus on HARP in our comparisons because it is designed for such capacity variations unlike most existing ML-based TE schemes [3]. We refer to the above adaptation of HARP for multiple classes as HARP-MC.

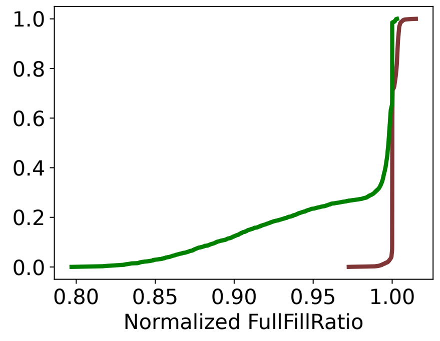
Figure 8 compares Hattrick and HARP-MC on the GEANT dataset. Figure 8(b) and (c) show that Hattrick significantly outperforms for medium and lower priority traffic. For example, Hattrick achieves 10%ile NormFulFill values of 1.0 and 0.9998 for the medium and lower priority classes, while the corresponding values for HARP-MC are 0.888 and 0.914. Further, Figure 8(a) shows these benefits of Hattrick do not come at the expense of higher priority traffic. Figure 9 shows Hattrick outperforms HARP-MC for PrivateWAN as well. There is a clear improvement in performance for low priority traffic and a noticeable improvement in the tail for medium priority traffic.

HARP-MC does not perform as well because of the residual approach – decisions are made for the higher priority traffic without considering the needs of other classes. In contrast, Hattrick fine-tunes routing of higher priority traffic to consider other classes while maintaining the objective of the higher priority class. The out-performance is stronger for GEANT since it is a more saturated network, and the need for such fine-tuning is more critical. Overall, the results show the benefits of Hattrick's alignment approach.

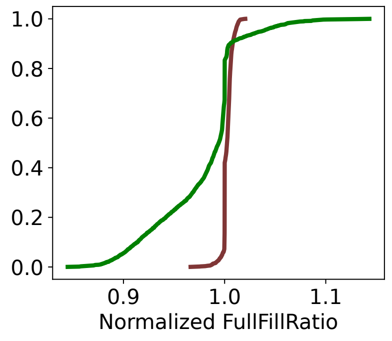
**Benefits of incorporating simulator into training.** As discussed in §3, Hattrick uses loss functions that consider both Fulfill-Ratio and MLU. Hattrick incorporates FulfillRatio by integrating a differentiable simulator in its training which models the effects of link capacity violations that may arise owing to prediction error. We



**(a) High traffic**

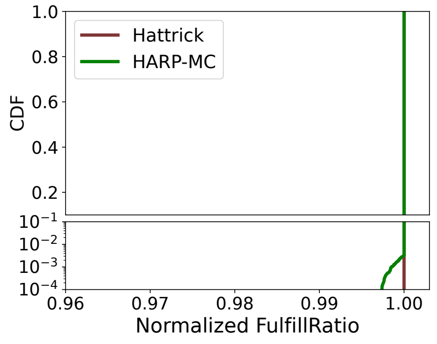


**(b) Medium traffic**

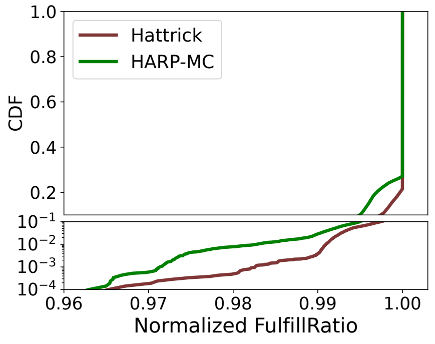


**(c) Low traffic**

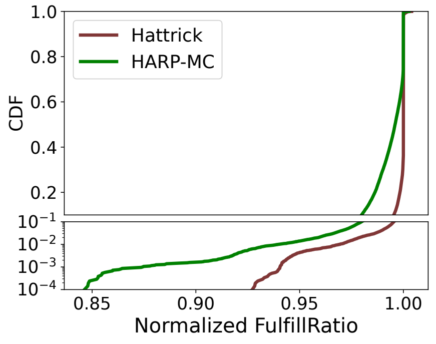
**Figure 8: Benefits of Hattrick's alignment approach. Comparing Hattrick (where routing of high priority traffic can be adjusted to accommodate lower priority traffic) to HARP-MC, which does not support such adjustment, using the GEANT dataset. Hattrick improves traffic of lower priority classes without compromising higher priority.**



**(a) High traffic**

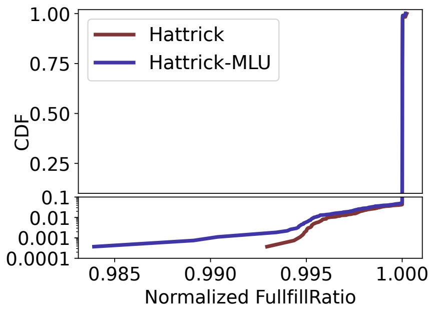


**(b) Medium traffic**

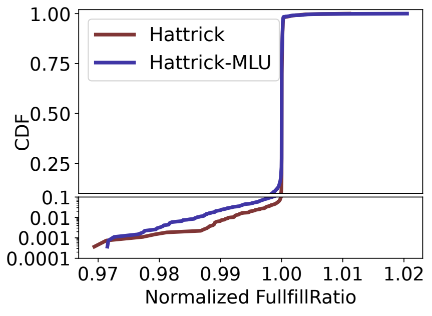


**(c) Low traffic**

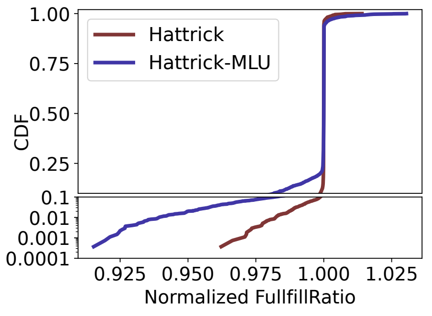
**Figure 9: Comparing Hattrick and HARP-MC for PrivateWAN.**



**(a) High traffic**



**(b) Medium traffic**



**(c) Low traffic**

**Figure 10: Benefits of Hattrick's approach of incorporating simulator into training compared to a variant Hattrick-MLU that does not since it only considers MLU metrics. Hattrick improves the performance of all classes but most clearly for low priority traffic.**

use GEANT dataset to evaluate the benefits of doing so by comparing Hattrick to a variant, which we call Hattrick-MLU which only considers MLU metrics and does not consider FulfillRatio. Figure 10 shows that Hattrick provides a noticeable improvement for low priority traffic, and a slight improvement for other classes. For example, the $1\%ile$ NormFulFill for Hattrick is 0.982 while only 0.940 for Hattrick-MLU. These results indicate the benefit integrating the simulator into the training process.

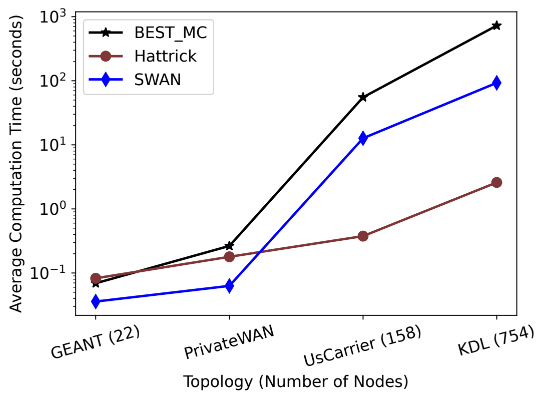


**Figure 11: Comparing computation times of schemes across different topologies.**

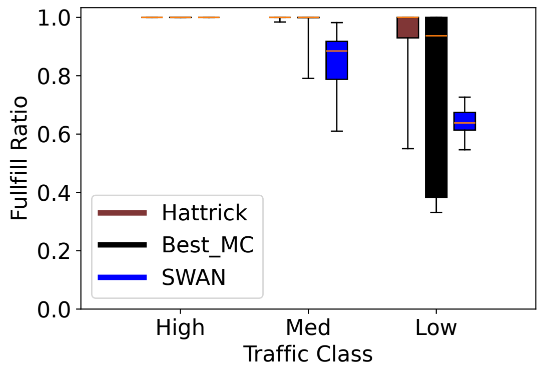


**Figure 12: Comparing Hattrick to BEST_MC and SWAN on UsCarrier.**

## 5.3 Benefits for larger topologies

Besides dealing better with prediction error, another potential advantage of ML-based schemes is their ability to reduce computation times for larger topologies. To investigate this, we compare Hattrick with BEST_MC and SWAN using two large topologies from the Internet Topology Zoo [22]. Specifically, we used UsCarrier (158 nodes, 378 edges) and KDL (754 nodes, 1790 edges). Since traffic matrices were not available, we used synthetic matrices via the code provided by [32] and randomly split the traffic into three classes by uniformly sampling high priority traffic to be between 0.45-0.5 of the total traffic, medium priority traffic to be between 0.4-0.45 of the total traffic, and the rest of the traffic is assigned to low priority traffic (0.05-0.15).

Figure 11 compares the computation time of Hattrick to SWAN and BEST_MC. The runtime for Hattrick was obtained through an NVIDIA RTX 4090 24GB GPU. SWAN and BEST_MC were implemented using Gurobi, and we report Gurobi runtime computed using an AMD EPYC 7763 64-Cores machine with 128GB RAM while using barrier methods without crossover with aggressive pre-solving.

The figure shows that for larger topologies, Hattrick significantly reduces computation time. For example, for UsCarrier, the average computation time per snapshot (three TMs) with Hattrick is 0.37 seconds, while the corresponding values for BEST_MC and SWAN are 55.20s and 12.68s respectively. For KDL, the time needed for Hattrick is 2.59 seconds on average, while BEST_MC and SWAN need 725.89 and 92.82 seconds respectively. Note that BEST_MC is more computationally expensive than SWAN because its second and third stage optimization problems have a much larger number of variables. For example, in the third stage it resolves for split ratios of all pairs and all classes, while SWAN only determines split ratios of traffic corresponding to the lowest priority class. For smaller topologies, all schemes have small and comparable computation times.

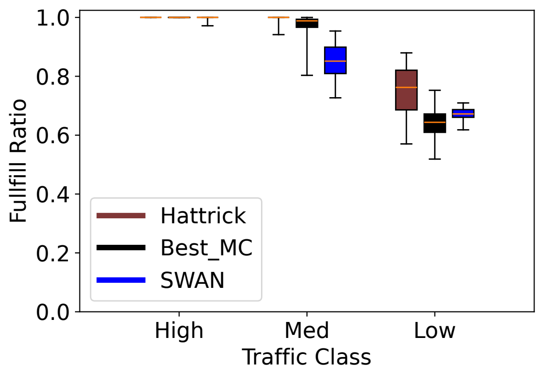


**Figure 13: Comparing Hattrick to BEST_MC and SWAN on KDL.**

Figure 12 and Figure 13 compare the performance of the three schemes for UsCarrier and KDL. Hattrick outperforms BEST_MC and SWAN for medium and low priority traffic, while the schemes perform comparably for higher priority traffic with SWAN doing slightly worse in the tail for KDL. Interestingly, when lower priority traffic is considered, SWAN achieves a higher median FulfillRatio than BEST_MC for KDL, and improves the tail for UsCarrier. . We believe this is because SWAN did not do as well on medium priority traffic, and could use the extra capacity for lower priority traffic. Overall, the results show Hattrick provides significant computation benefits for larger topologies while continuing to provide strong performance benefits.

## 5.4 Performance with other predictors

Our results so far have been presented using the exponential smoothing predictor. Figure 14 and 15 compare Hattrick with SWAN and BEST_MC using a linear regression predictor for the GEANT and PrivateWAN datasets respectively. Each plot presents three sets of boxplots (one for each priority class), with each set including one boxplot for each of the three scheme. The results are similar to the exponential smoothing predictor: (i) Hattrick outperforms BEST_MC and SWAN for both topologies; (ii) the performance wins are higher for GEANT, with a significant improvement in the median and $25^{th}$ %iles for all priority classes, and an even greater improvement in the tail; (iii) there is still a noticeable benefit in PrivateWAN especially with the lower priority class, while Hattrick improves the tail for higher priority classes. All of this confirms Hattrick continues to provide benefits with other predictors.

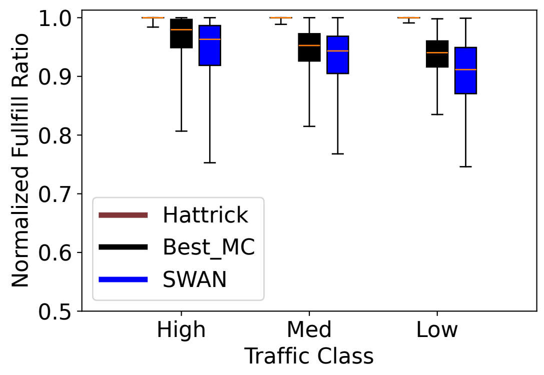


**Figure 14: Comparing Hattrick to other multiclass TE schemes using the linear regression predictor for GEANT.**

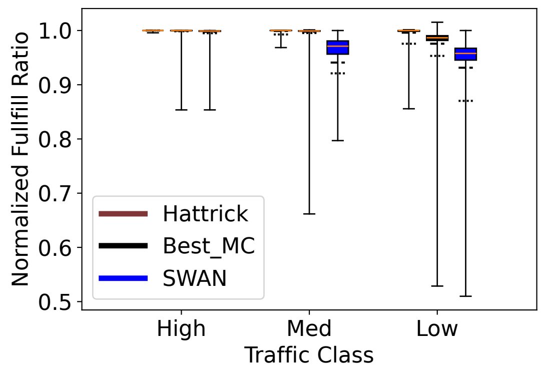


**Figure 15: Comparing Hattrick to other multiclass TE schemes using the linear regression predictor for Private-WAN. Dashed lines show the 10th and 1st percentiles respectively.**

## 6 Related work

Many recent works have explored ML techniques for TE [3, 14, 25, 32, 36, 40, 41, 44], and network planning [45]. We have discussed DOTE [32] extensively. Teal [40] uses a deep reinforcement learning approach. Teal focuses on demonstrating the computation benefits of ML-based TE for larger-scale topologies, and does not consider the impact of prediction error. HARP [3] was primarily designed to handle topology dynamics. None of these approaches consider multiple traffic classes, which is the focus of this paper.

Several works have used optimization-based methods for solving multiclass TE problems [15, 16, 18, 19, 23]. These works do not address issues related to prediction error, a key focus of this paper.

There is a rich literature of TE techniques that use oblivious routing and robust optimization methods to cope with demand prediction error [4, 5, 9, 10, 12, 13, 33, 37, 43]. These techniques focus on a single traffic class and typically on an MLU metric. Extending robust techniques to multiple traffic classes and other metrics such as Maxflow is an interesting area for future research. Further, robust optimization approaches requires explicitly specifying the scenarios being designed for, and can be overly conservative [7, 8, 19, 24]. Optimizing for expected performance (or other probabilistic measures) [7, 8, 19, 38] requires explicit probabilities of different scenarios, which is unfortunately challenging to get in practice. In contrast, ML-based TE can implictly learn probabilities and how to compensate for prediction error.

Recent works have been developing better predictors of traffic matrices using deep learning techniques [20, 27, 30, 39]. Incorporating better predictors into Hattrick can potentially further improve its performance.

Many ongoing efforts in both the networking [2, 15, 19, 28] and optimization communities [6] seek to scale optimization methods. ML methods such as Hattrick offer another approach to achieving such scaling. However, beyond computation benefits, Hattrick is motivated by the goal of achieving good performance under prediction error. Further, Hattrick can model feedback from simulators, an aspects that is hard to capture for traditional optimization techniques.

## 7 Conclusion

In this paper, we have presented Hattrick, the first ML-based TE approach for handling multiple traffic classes while optimizing their bandwidth requirements. As part of Hattrick we have developed (i) a novel neural architecture aligned with the sequence of optimization problems in multiclass TE; and (ii) a new precedence-aware multitask learning method tailored for multiclass TE given the need to optimize multiple metrics with a precedence relationship. Evaluations on a large private WAN and other public datasets show Hattrick outperforms state-of-the-art multiclass TE schemes including SWAN and BEST_MC by better coping with prediction error. For the GEANT network, Hattrick improves 1%ile NormFulFill (corresponding to the performance 2 9's of the time) relative to SWAN by 5.48% to 19.3% depending on the traffic class. Relative to BEST_MC, the corresponding improvements of Hattrick are 2.6%, 8% and 11% for the three traffic classes. Further, Hattrick reduces computation times by 280X and 36X relative to BEST_MC and SWAN respectively on a topology with 754 nodes.

**This work does not raise any ethical issues.**

**Acknowledgments.** We thank our shepherd, Dejan Kostic, and the anonymous reviewers for their feedback which helped to improve the paper. This work was funded in part by the National Science Foundation (NSF) award CNS-2212160 and CAREER IIS-1943364, and AFOSR FA95502210069.

## References

[1] GEANT network. http://geant3.archive.geant.net/Network/NetworkTopology/pages/home.aspx.

[2] Firas Abuzaid, Srikanth Kandula, Behnaz Arzani, Ishai Menache, Matei Zaharia, and Peter Bailis. Contracting wide-area network topologies to solve flow problems quickly. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 175–200. USENIX Association, April 2021.

[3] Abd AlRhman AlQiam, Yuanjun Yao, Zhaodong Wang, Satyajeet Singh Ahuja, Ying Zhang, Sanjay G. Rao, Bruno Ribeiro, and Mohit Tawarmalani. Transferable neural wan te for changing topologies. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, page 86–102, New York, NY, USA, 2024. Association for Computing Machinery.

[4] David Applegate, Lee Breslau, and Edith Cohen. Coping with network failures: Routing strategies for optimal demand oblivious restoration. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '04/Performance '04, pages 270–281, 2004.

[5] David Applegate and Edith Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proceedings of ACM SIGCOMM*, pages 313–324, 2003.

[6] David L. Applegate, Mateo D'iaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O'Donoghue, and Warren Schudy. Practical large-scale linear programming using primal-dual hybrid gradient. In *Neural Information Processing Systems*, 2021.

[7] Jeremy Bogle, Nikhil Bhatia, Manya Ghobadi, Ishai Menache, Nikolaj Bjorner, Asaf Valadarsky, and Michael Schapira. Teavar: Striking the right utilization-availability balance in wan traffic engineering. In *Proceedings of ACM SIGCOMM*, 2019.

[8] Yiyang Chang, Chuan Jiang, Ashish Chandra, Sanjay Rao, and Mohit Tawarmalani. Lancet: Better network resilience by designing for pruned failure sets. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3:1–26, 12 2019.

[9] Yiyang Chang, Sanjay Rao, and Mohit Tawarmalani. Robust validation of network designs under uncertain demands and failures. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 347–362, 2017.

[10] Chandra Chekuri. Routing and network design with robustness to changing or uncertain traffic demands. *SIGACT News*, 38(3):106–129, 2007.

[11] Marek Denis, Yuanjun Yao, Ashley Hatch, Qin Zhang, Chiun Lin Lim, Shuqiang Zhang, Kyle Sugrue, Henry Kwok, Mikel Jimenez Fernandez, Petr Lapukhov, Sandeep Hebbani, Gaya Nagarajan, Omar Baldonado, Lixin Gao, and Ying Zhang. Ebb: Reliable and evolvable express backbone network in meta. In *Proceedings of the ACM SIGCOMM 2023 Conference*, ACM SIGCOMM '23, page 346–359, New York, NY, USA, 2023. Association for Computing Machinery.

[12] N. G. Duffield, Pawan Goyal, Albert Greenberg, Partho Mishra, K. K. Ramakrishnan, and Jacobus E. van der Merive. A flexible model for resource management in virtual private networks. In *Proceedings of ACM SIGCOMM*, pages 95–108, 1999.

[13] Bernard Fortz and Mikkel Thorup. Robust optimization of OSPF/IS-IS weights. In *Proceedings of International Network Optimization Conference*, pages 225–230, 2003.

[14] Nan Geng, Mingwei Xu, Yuan Yang, Chenyi Liu, Jiahai Yang, Qi Li, and Shize Zhang. Distributed and adaptive traffic engineering with deep reinforcement learning. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, pages 1–10, 2021.

[15] A. Ghosh, Sangtae Ha, E. Crabbe, and J. Rexford. Scalable multi-class traffic management in data center backbone networks. *IEEE Journal on Selected Areas in Communications*, 31:2673–2684, 2013.

[16] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven wan. In *Proceedings of ACM SIGCOMM*, pages 15–26, 2013.

[17] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Kondapa Naidu B., Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, Kirill Mendelev, Steve Padgett, Faro Rabe, Saikat Ray, Malveeka Tewari, Matt Tierney, Monika Zahn, Jonathan Zolla, Joon Ong, and Amin Vahdat. B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 74–87, 2018.

[18] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. B4: Experience with a globally-deployed software defined wan. In *Proceedings of ACM SIGCOMM*, pages 3–14, 2013.

[19] Chuan Jiang, Zixuan Li, Sanjay Rao, and Mohit Tawarmalani. Flexile: Meeting bandwidth objectives almost always. In *Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '22, page 110–125, New York, NY, USA, 2022. Association for Computing Machinery.

[20] Grigorios Kakkavas, Michail Kalntis, Vasileios Karyotis, and Symeon Papavassiliou. Future network traffic matrix synthesis and estimation based on deep generative models. In *2021 International Conference on Computer Communications and Networks (ICCCN)*, pages 1–8, 2021.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[22] Simon Knight, Hung Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29:1765 – 1775, October 2011.

[23] Alok Kumar, Sushant Jain, Uday Naik, Nikhil Kasinadhuni, Enrique Cauich Zermeno, C. Stephen Gunn, Jing Ai, Björn Carlin, Mihai Amarandei-Stavila, Mathieu Robin, Aspi Siganporia, Stephen Stuart, and Amin Vahdat. Bwe: Flexible, hierarchical bandwidth allocation for wan distributed computing. In *Proceedings of ACM SIGCOMM*, 2015.

[24] Praveen Kumar, Yang Yuan, Chris Yu, Nate Foster, Robert Kleinberg, Petr Lapukhov, Chiun Lin Lim, and Robert Soulé. Semi-oblivious traffic engineering: The road not taken. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 157–170, 2018.

[25] Libin Liu, Li Chen, Hong Xu, and Hua Shao. Automated traffic engineering in sdwan: Beyond reinforcement learning. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 430–435, 2020.

[26] Ximeng Liu, Shizhen Zhao, Yong Cui, and Xinbing Wang. Figret: Fine-grained robustness-enhanced traffic engineering. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, page 117–135, New York, NY, USA, 2024. Association for Computing Machinery.

[27] T. Mallick, M. Kiran, B. Mohammed, and P. Balaprakash. Dynamic graph neural network for traffic forecasting in wide area networks. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1–10, Los Alamitos, CA, USA, dec 2020. IEEE Computer Society.

[28] Pooria Namyar, Behnaz Arzani, Srikanth Kandula, Santiago Segarra, Daniel Crankshaw, Umesh Krishnaswamy, Ramesh Govindan, and Himanshu Raj. Solving Max-Min fair resource allocations quickly on large graphs. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 1937–1958, Santa Clara, CA, April 2024. USENIX Association.

[29] Aviv Navon, Idan Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. In *International Conference on Machine Learning (ICML)*, 2022.

[30] Laisen Nie, Dingde Jiang, Lei Guo, and Shui Yu. Traffic matrix prediction and estimation based on deep learning in large-scale ip backbone networks. *Journal of Network and Computer Applications*, 76:16–22, 2016.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[32] Yarin Perry, Felipe Vieira Frujeri, Chaim Hoch, Srikanth Kandula, Ishai Menache, Michael Schapira, and Aviv Tamar. DOTE: Rethinking (predictive) WAN traffic engineering. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1557–1581, Boston, MA, April 2023. USENIX Association.

[33] Michal Pióro and Deepankar Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[34] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[35] Steve Uhlig, Bruno Quoitin, Jean Lepropre, and Simon Balon. Providing public intradomain traffic matrices to the research community. *SIGCOMM Comput. Commun. Rev.*, 36(1):83–86, jan 2006.

[36] Asaf Valadarsky, Michael Schapira, Dafna Shahaf, and Aviv Tamar. Learning to route. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, HotNets '17, page 185–191, New York, NY, USA, 2017. Association for Computing Machinery.

[37] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. COPE: Traffic engineering in dynamic networks. In *Proceedings of ACM SIGCOMM*, pages 99–110, 2006.

[38] Yiting Xia, Ying Zhang, Zhizhen Zhong, Guanqing Yan, Chiun Lin Lim, Satyajeet Singh Ahuja, Soshant Bali, Alexander Nikolaidis, Kimia Ghobadi, and Manya Ghobadi. A social network under social distancing: Risk-Driven backbone management during COVID-19 and beyond. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 217–231. USENIX Association, April 2021.

[39] Shenghe Xu, Murali Kodialam, T. V. Lakshman, and Shivendra S. Panwar. Learning based methods for traffic matrix estimation from link measurements. *IEEE Open Journal of the Communications Society*, 2:488–499, 2021.

[40] Zhiying Xu, Francis Y. Yan, Rachee Singh, Justin T. Chiu, Alexander M. Rush, and Minlan Yu. Teal: Learning-accelerated optimization of wan traffic engineering. In *Proceedings of the ACM SIGCOMM 2023 Conference*, ACM SIGCOMM '23, page 378–393, New York, NY, USA, 2023. Association for Computing Machinery.

[41] Zhiyuan Xu, Jian Tang, Jingsong Meng, Weiyi Zhang, Yanzhi Wang, Chi Harold Liu, and Dejun Yang. Experience-driven networking: A deep reinforcement learning based approach. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, page 1871–1879. IEEE Press, 2018.

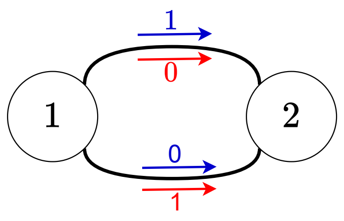
[42] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[43] C. Zhang, Zihui Ge, J. Kurose, Y. Liu, and D. Towsley. Optimal routing with multiple traffic matrices tradeoff between average and worst case performance. In *Network Protocols, 2005. ICNP 2005. 13th IEEE International Conference on*, 2005.

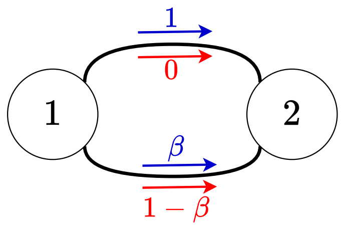
[44] Junjie Zhang, Minghao Ye, Zehua Guo, Chen-Yu Yen, and H. Jonathan Chao. Cfr-rl: Traffic engineering with reinforcement learning in sdn. *IEEE Journal on Selected Areas in Communications*, 38(10):2249–2259, 2020.

[45] Hang Zhu, Varun Gupta, Satyajeet Singh Ahuja, Yuandong Tian, Ying Zhang, and Xin Jin. Network planning with deep reinforcement learning. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 258–271, New York, NY, USA, 2021. Association for Computing Machinery.

| | High | Low |
|---|---|---|
| Actual | $1+\beta$ | 1 |
| Predicted | 1 | 1 |



**(a) BEST_MC when provided with predicted demands**



**(b) BEST_MC when provided with actual demands by an oracle.**

**Figure 16: An illustration of how even the optimal strategy (BEST_MC) can result in a priority inversion owing to inaccuracies in predicted demand.**

## A Appendix A

Appendices are supporting material that has not been peer-reviewed.

### A.1 Priority inversion with BEST_MC under prediction errors.

As discussed earlier, a sub-optimal scheme may achieve a NormFulFill higher than 1 for lower classes because of a priority inversion – i.e., it may not transmit as much high priority traffic as the optimal strategy, and may use the additional capacity to transmit more lower priority traffic. In this section, we illustrate that (i) a priority inversion could result even when the optimal strategy (BEST_MC) is used on a *predicted* matrix; and (ii) the inversion could be significant depending on the error.

Consider Figure 16 which depicts a simple two node network connected by two links, each with a capacity of 1 unit. There are two priority classes, and the predicted traffic demand involves 1 unit of each of high and low priority traffic. However, the true traffic demand involves $1 + \beta$ units of high traffic and 1 unit of low traffic.

When provided the predicted demand, a potential solution for BEST_MC is to route 100% of high priority traffic on the upper link, and 100% of the low priority traffic on the lower link. When these split ratios are applied to the true demand, the scheme can effectively only carry 1 unit of high priority traffic and 1 unit of low priority traffic as shown in Figure 16(a). In contrast, the optimal strategy when provided the true demand by an oracle is to transmit the entire $1+\beta$ units of high priority traffic and use the residual capacity of $1-\beta$ for low priority traffic (Figure 16(b) shows a potential solution). Thus, BEST_MC when provided the predicted demand would achieve a NormFulFill$^{high}$ of $\frac{1}{1+\beta}$ and a NormFulFill$^{low}$ of $\frac{1}{1-\beta}$. The NormFulFill of low priority not only exceeds 1, but can be arbitrarily large based on $\beta$.

## A.2 Mathematical Formulation of BEST_MC

The mathematical formulation outlined in Function 2 describes the three-stage optimization problem solved by BEST_MC. Each stage incrementally incorporates a new traffic class while preserving the total amount of traffic fulfilled in previous stages.

- **Stage 1:** Optimizes high-priority traffic ($c = h$) by maximizing the total fulfilled demand, subject to demand and link capacity constraints.
- **Stage 2:** Adds medium-priority traffic ($c \in \{h, m\}$) while preserving the total high-priority fulfillment obtained in Stage 1.
- **Stage 3:** Adds low-priority traffic ($c \in \{h, m, l\}$) while preserving both the high-priority fulfillment from Stage 1 and the combined high + medium fulfillment from Stage 2.

The decision variables $w^c_{stk,i}$ represent the fraction of class $c$ demand from source $s$ to destination $t$ routed through tunnel $k$ at stage $i$.

**Function 2** BEST_MC Mathematical Formulation

$$\begin{aligned}
\text{maximize}\quad & F^h = \sum_{s,t,k} w^h_{stk,1} \cdot d^h_{st} \\
\text{subject to}\quad & \sum_k w^1_{stk,1} \le 1 && \forall (s,t) \in N \\
& \sum_{s,t,k} w^h_{stk,1} \cdot d^h_{st} \cdot T_{ijstk} \le C_{ij} && \forall (i,j) \in E \\
& w^h_{stk,1} \ge 0 && \forall (s,t,k)
\end{aligned}$$

Denote optimal objective as $\tilde{F}^h$

$$\begin{aligned}
\text{maximize}\quad & F^{hm} = \sum_{c\in\{h,m\}} \sum_{s,t,k} w^c_{stk,2} \cdot d^c_{st} \\
\text{subject to}\quad & \sum_k w^c_{stk,2} \le 1 && \forall (s,t) \in N,\ c \in \{h,m\} \\
& \sum_{c\in\{h,m\}} \sum_{s,t,k} w^c_{stk,2} \cdot d^c_{st} \cdot T_{ijstk} \le C_{ij} && \forall (i,j) \in E \\
& \sum_{s,t,k} w^h_{stk,2} \cdot d^h_{st} = \tilde{F}^h \\
& w^c_{stk,2} \ge 0 && \forall (s,t,k),\ c \in \{h,m\}
\end{aligned}$$

Denote optimal objective as $\tilde{F}^{hm}$

$$\begin{aligned}
\text{maximize}\quad & F^{hml} = \sum_{c\in\{h,m,l\}} \sum_{s,t,k} w^c_{stk,3} \cdot d^c_{st} \\
\text{subject to}\quad & \sum_k w^c_{stk,3} \le 1 && \forall (s,t) \in N,\ c \in \{h,m,l\} \\
& \sum_{c\in\{h,m,l\}} \sum_{s,t,k} w^c_{stk,3} \cdot d^c_{st} \cdot T_{ijstk} \le C_{ij} && \forall (i,j) \in E \\
& \sum_{s,t,k} w^h_{stk,3} \cdot d^h_{st} = \tilde{F}^h \\
& \sum_{c\in\{h,m\}} \sum_{s,t,k} w^c_{stk,3} \cdot d^c_{st} = \tilde{F}^{hm} \\
& w^c_{stk,3} \ge 0 && \forall (s,t,k),\ c \in \{h,m,l\}
\end{aligned}$$

**Where:**

(1) $E$: Set of all links/edges, $\langle i,j\rangle \in E$
(2) $C_{ij}$: Capacity of link $\langle i,j\rangle$
(3) $d^c_{st}$: Traffic of class $c$ from source $s$ to destination $t$
(4) $N$: Set of all source-destination node pairs, $\langle s,t\rangle \in N$
(5) $T_{ijstk}$: Binary indicator for whether link $\langle i,j\rangle$ is used in tunnel $k$ from $s$ to $t$
(6) $w^c_{stk,i}$: Split ratio of $d^c_{st}$ from source $s$ to destination $t$ on tunnel $k$ at stage $i$